MASTER

PI-MBUS-300 Rev B

Gould
Modbus Protocol

REFERENCE GUIDE

SUBJECT: This document contains a description of the Modbus protocol, a detailed explanation of Modbus functions, and other information pertinent to using Gould Programmable Controllers in a Modbus system.

January, 1985

Gould Inc.
Programmable Control Division
P.O. Box 3083
Andover, Massachusetts 01810

PREFACE

This reference guide describes an industrial communications and distributed control system — the Modbus Communications system. This system integrates PC's, computers, terminals, and other monitoring, sensing, and control devices.

Modbus systems have been installed for energy management, transfer line control, and pipeline monitoring. The system's reliability and flexibility allow it to handle all kinds of processes and operations in just about any industry. Like our PC's, it's rugged enough for an industrial environment. Since this communications system is specifically designed for PC's, it's as flexible in its applications as they are. The Modbus Communications system is Gould's commitment to provide industrial communications tools for the future.

Related Documents:

Modbus 184/384 Programming Protocol	PI-MBUS-301
Modbus 484 Programming Protocol	PI-MBUS-302
Modbus 584 Programming Protocol	PI-MBUS-303
Modbus System Planning	ML-MBUS-PLN

Printed in U.S.A.

GOULD INC., PROGRAMMABLE CONTROL DIVISION
CONFIDENTIALITY AGREEMENT

Gould Inc., Programmable Control Division (hereinafter referred to as "Gould") and the user's of this document (hereinafter referred to as "The Company") desire to share proprietary technical information concerning Gould's Modbus communications system.

For this reason Gould is disclosing to The Company information in the form of a Modbus Protocol Reference Guide. In as much as Gould considers this information to be proprietary and desires that it be maintained in confidence, it is hereby agreed by The Company that such data shall be maintained in confidence by The Company for a period of ten years following the date of this agreement.

During this period, The Company shall not divulge such information to any third party without the prior written consent of Gould and shall take reasonable efforts to prevent any unauthorized disclosure by its employees. However, The Company shall not be obligated to keep such information in confidence if it was in The Company's possession prior to its receipt from Gould, if it is or becomes public knowledge without the fault of The Company, or the information becomes available on an unrestricted basis from a third party having a legal right to disclose such information.

The Company's receipt of the Modbus Protocol Reference Guide constitutes acceptance of these terms.

TABLE OF CONTENTS

CONTENTS (cont.)

FIGURES

CONTENTS (cont.)

TABLES

SECTION 1
MODBUS PROTOCOL

A data communications system protocol controls the language structure or message format common to all devices on a network. The protocol is vital to the system's operation; it determines how the master and slaves establish and break off contact, how the sender and receiver are identified, how messages are exchanged in an orderly manner, and how errors are detected. The protocol controls the query and response cycle which takes place between master and slave devices, as shown in Figure 1-1.

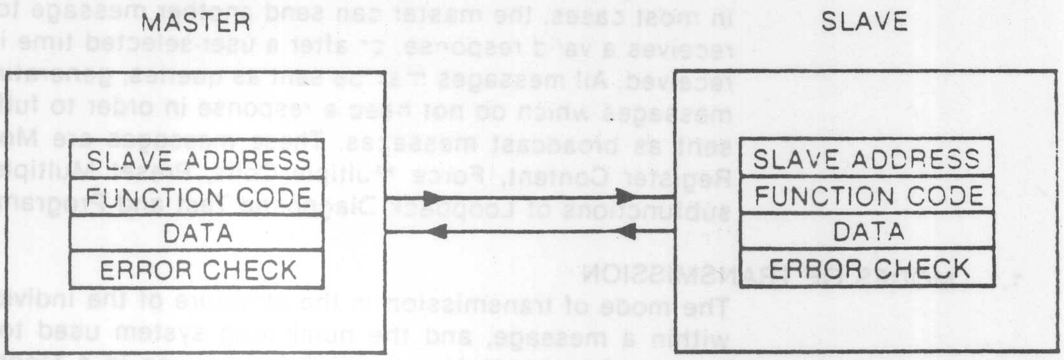

Figure 1-1. Master and Slave Query/Response Cycle

The protocol provides for one master and up to 247 PC slaves on a common line. Although the protocol supports up to 247 slaves, certain device restrictions may limit the number of slaves to a number less than 247. For example, Gould's J478 modem restricts the total number of slaves on the line to 32. Each slave is assigned a fixed unique device address in the range of 1 to 247.

Only the master initiates a transaction. Transactions are either a query/response type (only a single slave is addressed) or a broadcast/no response type (all slaves are addressed). A transaction comprises a single query and single response frame or a single broadcast frame.

Certain characteristics of the Modbus protocol are fixed, such as the frame format, frame sequences, handling of communications errors and exception conditions, and the functions performed.

Other characteristics are user selectable. These include a choice of transmission medium, baud rate, character parity, number of stop bits, and transmission modes (ASCII or RTU). The user selected parameters are set (hardwired or programmed) at each station. These parameters cannot be changed while the system is running.

In order for hardware to send messages over data lines, the message must be contained in an envelope. The envelope leaves the hardware through a "port" and is "carried" over the line to the addressed device. In this case, the Modbus protocol provides the envelope in the form of message frames. The information in the message is the address of the intended receiver, what the receiver is to do, data needed to perform an action, and a means of checking for errors.

When the message reaches the Modbus Slave Interface, it enters that addressed device through a similar "port". The addressed device removes the envelope, reads the message, and, if no errors have occurred, performs the requested task. It then replaces the message into the saved envelope and "returns to sender". The information in the response message is the slave address, the action performed, data acquired as a result of the action, and a means of checking for errors. No response is transmitted if the message is a "broadcast", (a message to all slaves) as indicated by an address of 0.

The receiving device "writes" and sends a response of its own if an erroneous message is received. The functions that a Modbus slave interface can perform are defined in Tables B1, B2, and B3 for the 1/384, 484 and 584 respectively. (See Appendix B). Also see Table 3-1.

In most cases, the master can send another message to any slave as soon as it receives a valid response, or after a user-selected time interval if no response is received. All messages may be sent as queries, generating slave responses. Only messages which do not need a response in order to fulfill their function may be sent as broadcast messages. These messages are Modify Coil Status, Modify Register Content, Force Multiple Coils, Preset Multiple Registers, and certain subfunctions of Loopback Diagnostic Test and Program.

1.1 MODES OF TRANSMISSION

The mode of transmission is the structure of the individual units of information within a message, and the numbering system used to transmit the data. Two modes of transmission are available for use in a Modbus system. Both modes provide the same capabilities for communicating with PC slaves; the mode is selected depending on the equipment used as a Modbus master. One mode must be used per Modbus system: mixing of modes is not allowed. The modes are ASCII (American Standard Code for Information Interchange), and RTU (Remote Terminal Unit). The modes are defined in Table 1-1.

Table 1-1. Characteristics of ASCII and RTU Modes of Transmission

Characteristic	ASCII (7-bit)	RTU (8-bit)
Coding System	hexadecimal (uses ASCII printable characters: 0-9,A-F)	8-bit binary
Number of bits per character:		
start bits	1	1
data bits (least significant first)	7	8
parity (optional)	1 (1-bit sent for even or odd parity, no bits for no parity)	1 (1-bit sent for even or odd parity, no bits for no parity)
stop bits	1 or 2	1 or 2
Error checking	LRC (Longitudinal Redundancy Check) See paragraph 1.2.2	CRC (Cyclical Redundancy Check) See paragraph 1.2.1

ASCII printable characters are easy to view when troubleshooting and this mode is suited to computer masters programmed in a high level language, such as FORTRAN, as well as PC masters. RTU is suited to computer masters programmed in a machine language, as well as PC masters.

In the RTU mode, data is sent in 8-bit binary characters. In the ASCII mode, each RTU character is first divided into two 4-bit parts (high order and low order) and then represented by their hexadecimal equivalent. The ASCII characters representing the hexidecimal characters are used to construct the message. The ASCII mode uses twice as many characters as the RTU mode, but decoding and handling the ASCII data is easier. Additionally, in the RTU mode, message characters must be transmitted in a continuous stream. In the ASCII mode, breaks of up to 1 second can occur between characters to allow for a relatively slower master.

1.2 ERROR DETECTION

There are two types of errors which may occur in a communications system: transmission errors and programming or operation errors. The Modbus system has specific methods for dealing with either type of error.

Communications errors usually consist of a changed bit or bits within a message. For example, 0001 0100 may become 0001 0110, changing the decimal value from 20 to 22. It is much less common to have a bit added to or deleted from a message. The most frequent cause of communications errors is noise: unwanted electrical signals in a communications channel. These signals occur because of electrical interference from machinery, damage to the communications channel, impulse noise (spikes), etc. Communications errors are detected by character framing, a parity check, and a redundancy check.

When the character framing, parity, or redundancy checks detect a communications error, processing of the message stops. A PC slave will not act on or respond to the message. (The same result occurs if a non-existent slave address is used.)

When a communication error occurs, the message is unreliable. The PC slave cannot know for sure if this message was intended for it. So the CPU might be answering a message which was not its message to begin with. It is essential to program the Modbus master to assume a communications error has occurred if there is no response in a reasonable time. The length of this time period depends upon baud rate, type of message, and scan time of the PC slave. Once this time is determined, the master may be programmed to automatically retransmit the message.

Both modes of transmission, RTU and ASCII, include an optional parity bit in their character format. In RTU mode, it is the ninth bit of the data field (8 bits of data and the parity bit). In ASCII mode it is the eighth bit of the data field (7 bits of data and the parity bit). If parity is not used, the parity bit is not transmitted. Parity is optional for the Modbus system; the interface units are configured for no parity, even parity, or odd parity at the time of installation. All units on a system must be configured for the same option.

Parity helps detect single bit communications errors. The Modbus system determines whether the parity bit should be a 1 or 0 in the following manner:

- add the number of 1's in the data
- determine if the number is even or odd

For even parity:

- if the number is even, add a 0 as the parity bit to keep the number of 1's even
- if the number is odd, add a 1 as the parity bit to make the number of 1's even
- for example, in even parity the data 0110 1000 would have a 1 as the parity bit, while the data 0110 1010 would have a zero as the parity bit

For odd parity:

- the master adds a 1 or zero as the parity bit so that the number of 1's in the data is odd

If there are two errors within a message, however, parity may not be able to detect the changes. If 0010 0000 is distorted to 0010 0011, the number of 1's in the data is still odd.

The Modbus system provides several levels of error checking in order to assure the quality of the data transmission. To detect multibit errors where the parity has not changed, the system uses redundancy checks: Cyclical Redundancy Check (CRC) and Longitudinal Redundancy Check (LRC). Which redundancy check is used is dependent upon the mode of transmission. RTU uses the cyclical check and ASCII uses the longitudinal check. The generation of CRC and LRC is explained below. The CRC and LRC error checks are performed automatically. Paragraphs 1.2.1 and 1.2.2 are included for your information only.

1.2.1 CRC-16 (Cyclic Redundancy Check) Error Check Sequence

The CRC-16 error check sequence is implemented as described in the following paragraphs.

The message (data bits only, disregarding start/stop and optional parity bits) is considered as one continuous binary number whose most significant bit (MSB) is transmitted first. The message is pre-multiplied by x^{16} (shifted left 16 bits), then divided by $x^{16} + x^{15} + x^2 + 1$ expressed as a binary number (11000000000000101). The integer quotient digits are ignored and the 16-bit remainder (initialized to all ones at the start to avoid the case of all zeros being an accepted message) is appended to the message (MSB first) as the two CRC check bytes. The resulting message including CRC, when divided by the same polynomial ($x^{16} + x^{15} + x^2 + 1$) at the receiver will give a zero remainder if no errors have occurred. (The receiving unit recalculates the CRC and compares it to the transmitted CRC). All arithmetic is performed modulo two (no carries). An example of the CRC-16 error check for message HEX 0207 (address 2, function 7 or a status request to slave number 2) is given in Figure 1-2.

The device used to serialize the data for transmission will send the conventional LSB or right-most bit of each character first. In generating the CRC, the first bit transmitted is defined as the MSB of the dividend. For convenience then, and since there are no carries used in arithmetic, let's assume while computing the CRC that the MSB is on the right. To be consistent, the bit order of the generating polynomial must be reversed. The MSB of the polynomial is dropped since it affects only the quotient and not the remainder. This yields 1010 0000 0000 0001 (Hex A001). Note that this reversal of the bit order will have no affect whatever on the interpretation or bit order of characters external to the CRC calculations.

The step by step procedure to form the CRC-16 check bytes is as follows:

1. Load a 16-bit register with all 1's.

2. Exclusive OR the first 8-bit byte with the high order byte of the 16-bit register, putting the result in the 16-bit register.

3. Shift the 16-bit register one bit to the right.

4a. If the bit shifted out to the right (flag) is one, exclusive OR the generating polynomial 1010 000 000 0001 with the 16-bit register.

4b. If the bit shifted out to the right is a zero; return to step 3.

5. Repeat steps 3 and 4 until 8 shifts have been performed.

6. Exclusive OR the next 8-bit byte with the 16-bit register.

7. Repeat step 3 through 6 until all bytes of the message have been exclusive OR with the 16-bit register and shifted 8 times.

6 2 07

8. The contents of the 16-bit register are the 2 byte CRC error check and is added to the message most significant bits first.

	16-BIT REGISTER			MSB	FLAG
(Exclusive or)	1111	1111	1111	1111	
0			0000	0010	
	1111	1111	1111	1101	
Shift 1	0111	1111	1111	1110	1
Polynomial	1010	0000	0000	0001	
	1101	1111	1111	1111	
Shift 2	0110	1111	1111	1111	1
Polynomial	0101	0000	0000	0001	
	1100	1111	1111	1110	
Shift 3	0110	0111	1111	1111	0
Shift 4	0011	0011	1111	1111	1
Polynomial	1010	0000	0000	0001	

Figure 1-2. Example — CRC-16 Generation — Read Exception Status of Slave 02

	16-BIT REGISTER			MSB	FLAG
	1001	0011	1111	1110	
Shift 5	0100	1001	1111	1111	0
Shift 6	0010	0100	1111	1111	1
Polynomial	1010	0000	0000	0001	
	1000	0100	1111	1110	
Shift 7	0100	0010	0111	1111	0
Shift 8	0010	0001	0011	1111	1
Polynomial	1010	0000	0000	0001	
	1000	0001	0011	1110	
07			0000	0111	
	1000	0001	0011	1001	
Shift 1	0100	0000	1001	1100	1
Polynomial	1010	0000	0000	0001	
	1110	0000	1001	1101	
Shift 2	0111	0000	0100	1110	1
Polynomial	1010	0000	0000	0001	
	1101	0000	0100	1111	
Shift 3	0110	1000	0010	0111	1
Polynomial	1010	0000	0000	0001	
	1100	1000	0010	0110	
Shift 4	0110	0100	0001	0011	0
Shift 5	0011	0010	0000	1001	1
Polynomial	1010	0000	0000	0001	
	1001	0010	0000	1000	
Shift 6	0100	1001	0000	0100	0
Shift 7	0010	0100	1000	0010	0
Shift 8	0001	0010	0100	0001	0
		HEX 12		HEX 41	

TRANSMITTED MESSAGE WITH CRC-16
(MESSAGE SHIFTED TO RIGHT TO TRANSMIT)

12		41		07		02	
0001	0010	0100	0001	0000	0111	0000	0010

Last bit Transmitted — TRANSMISSION ORDER — First Bit Transmitted

Figure 1-2. Example — CRC-16 Generation — Read Exception Status of Slave 02 (cont)

1.2.2 LRC (Longitudinal Redundancy Check) Error Check Sequence

The error check sequence for the ASCII mode is LRC. The error check is an 8-bit binary number represented and transmitted as two ASCII hexadecimal (hex) characters. The error check is produced by converting the hex characters to binary, adding the binary characters without wraparound carry, and two's complementing the result. (See Figure 1-3.) At the received end the LRC is recalculated and compared to the sent LRC. The colon, CR, LF, and any imbedded non-ASCII hex characters are ignored in calculating the LRC.

Address	0	2	0000	0010
Function	0	1	0000	0001
Start Add H.O.	0	0	0000	0000
Start Add L.O.	0	0	0000	0000
Quantity of Pts	0	0	0000	0000
	0	8	0000	1000
			0000	1011
1's Complement			1111	0100
+1				1
2's Complement			1111	0101
Error Check	F	5	F	5

Receiving PC sums up all data bytes received including the LRC at the end. The 8-bit should all equal zero. (Note, the sum may exceed 8 bits — only the low order of bits should be saved.)

	0000	0010
	0000	0001
	0000	0000
	0000	0000
	0000	0000
OK	0000	1000
Error Check	1111	0101
SUM	0000	0000

Figure 1-3. EXAMPLE — LRC Generation — Read First 8 Coils From Slave 02

1.3 MODBUS PROTOCOL

As shown in the protocol overview, protocol covers the rules for communication between a master and slaves. With a PC master, the Modbus protocol is incorporated into the PC interface, and all communication is transparent. (The Modbus Communications Handler is a FORTRAN-based software product which programs a computer master to handle the Modbus system protocol. In this case, communications to a PC slave on a Modbus system are transparent to user-developed applications programs). The Modbus protocol is tailored to the type of communications that an industrial user needs for a network of PC's. In general the interpretation of fields within a message are identical between RTU and ASCII transmission modes. (See Figures 1-4, and 1-5.) The major differences are the type of error check performed on the message and that approximately twice as many characters are used in ASCII. Instead of sending a single 8-bit binary character the equivalent pair of 7-bit ASCII (0-9,A-0F) characters are sent.

1.3.1 ASCII Framing

Framing in ASCII transmission mode is accomplished by the use of the unique colon(:) character to indicate beginning of frame and carriage return (CR) line feed (LF) to delineate end of frame. The line feed character also serves as a synchronizing character which indicates that the transmitting station is ready to receive an immediate reply. See Figure 1-4.

BEG OF FRAME	ADDRESS	FUNCTION	DATA	ERROR CHECK	EOF	READY TO REC RESP
:	2-CHAR 16-BITS	2-CHAR 16-BITS	N X 4-CHAR N X 16-BITS	2-CHAR 16-BITS	CR	LF

CHAR = CHARACTER: 1 CHARACTER = 7 DATA BITS, 1 START BIT, 1 OR 2 STOP BITS AND OPTIONALLY — 1 PARITY BIT

Figure 1-4. ASCII Message Frame Format

1.3.2 Remote Terminal Unit (RTU) Framing

Frame synchronization can be maintained in RTU transmission mode only by simulating a synchronous message. The receiving device monitors the elapsed time between receipt of characters. If three and one-half character times elapse without a new character or completion of the frame, then the device flushes the frame and assumes that the next byte received will be an address. See Figure 1-5.

T1 T2 T3	ADDRESS	FUNCTION	DATA	CHECK	T1 T2 T3
	8-BITS	8-BITS	N X 8-BITS	16-BITS	

Figure 1-5. RTU Message Frame Format

1.3.3 Address Field

The address field immediately follows the beginning of frame and consists of 8-bits (RTU) or 2 characters (ASCII). These bits indicate the user assigned address of the slave device that is to receive the message sent by the attached master.

Each slave must be assigned a unique address and only the addressed slave will respond to a query that contains its address. When the slave sends a response, the slave address informs the master which slave is communicating. In a broadcast message, an address of 0 is used. All slaves interpret this as an instruction to read and take action on the message, but not to issue a response message.

1.3.4 Function Field

The function code field tells the addressed slaves what function to perform. Modbus function codes are specifically designed for interacting with a PC on the Modbus industrial communications system. Table 1-2 lists the function codes, their meaning, and the action they initiate.

The high order bit in this field is set by the slave device to indicate that other than a normal response is being transmitted to the Master device. (See Section 2 for a description of exception responses.) This bit remains 0 if the message is a query or a normal response message.

Table 1-2. Modbus Function Codes

CODE	MEANING	ACTION
01	READ COIL STATUS	Obtains current status (ON/OFF) of a group of logic coils.
02	READ INPUT STATUS	Obtain current status (ON/OFF) of a group of discrete inputs.
03	READ HOLDING REGISTERS	Obtain current binary value in one or more holding registers.
04	READ INPUT REGISTERS	Obtain current binary value in one or more input registers.
05	FORCE SINGLE COIL	Force logic coil to a state of ON or OFF.
06	PRESET SINGLE REGISTER	Place a specific binary value into a holding register.
07	READ EXCEPTION STATUS	Obtain the status (ON/OFF) of the eight internal coils whose addresses are controller dependent (see Section 3.7). User logic can program these coils to indicate slave status. Short message length allows rapid reading of status.
08	LOOPBACK DIAGNOSTIC TEST	Diagnostic test message sent to slave to evaluate communications processing.
09	PROGRAM (484 ONLY)	Allows master to simulate actions of programming panel and alter PC slave logic.
10	POLL PROGRAM COMPLETE (484 ONLY)	Allows master to communicate with other slaves if one slave is working on lengthy program task. Slave is polled periodically to see if its program operation is complete. Only issued after a message containing function Code 9 has been sent.

Table 1-2. Modbus Function Codes (cont.)

CODE	MEANING	ACTION
11	FETCH EVENT COUNTER COMMUNICATIONS	Allows a master to issue a single query and subsequently determine whether the operation was successfully performed, especially when a communication error occurred on that command or its response.
12	FETCH COMMUNICATIONS EVENT LOG	Allows a master to retrieve a communications event log that contains information about each Modbus transaction with that slave. If a transaction was not completed, the log provides information about the error that occurred.
13	PROGRAM (184/384, 484, 584)	Allows master to simulate actions of programming panel and alter PC slave logic.
14	POLL PROGRAM COMPLETE (184/384, 484, 584)	Allows master to communicate with other slaves if one slave is working on lengthy program task. Slave is polled periodically to see if its program operation is complete. Only issued after a message containing a function Code 13 has been sent.
15	FORCE MULTIPLE COILS	Forces a series of consecutive logic coils to defined ON or OFF states.
16	PRESET MULITPLE REGISTERS	Places specific binary values into a series of consecutive holding registers.
17	REPORT SLAVE I.D.	Allows a master to determine the type of slave addressed and the status of the slave's run light.
18	PROGRAM (584 & Micro 84)	Allows Master to simulate actions of programming panel and alter PC state logic.
19	RESET COMMUNICATIONS LINK	Resets slave to known state after non-recoverable error. Resets sequence byte.
20	READ GENERAL REFERENCE (584L ONLY)	Displays information contained in extended memory files.
21	WRITE GENERAL REFERENCE (584L ONLY)	Enters or changes information contained in extended memory files.
22-64	RESERVED FOR EXPANDED FUNCTIONS	

Table 1-2. Modbus Function Codes (cont.)

CODE	MEANING	ACTION
65-72	RESERVED FOR USER FUNCTIONS	Reserved for users for custom functions; will not be used by Gould in future products.
73-119	ILLEGAL FUNCTIONS	
120-127	RESERVED	Reserved for internal use.
128-255	RESERVED	Reserved for exception responses.

1.3.5 Data Field

The data field contains information needed by the slave to perform the specific function or it contains data collected by the slave in response to a query. This information may be values, address references, or limits. For example, the function code tells the slave to read a holding register, and the data field is needed to indicate which register to start at and how many to read. The imbedded address and data information varies with the type and capacity of PC associated with the slave.

1.3.6 Error Check Field

This field allows the master and slave devices to check a message for errors in transmission. Sometimes, because of electrical noise or other interference, a message may be changed slightly while it is on its way from one unit to another. The error checking assures that the slave or master does not react to messages that have changed during transmission. This increases the safety and the efficiency of the Modbus system.

The error check field uses a longitudinal redundancy check (LRC) in the ASCII Mode, and a CRC-16 check in the RTU Mode. (See paragraph 1.2 for a description of the two types of error check.)

If query and response messages could be read in English as they were transmitted, the four fields of the messages would look like they do in Figure 1-6. (Note that the sending sequence is always the same — Address, Function Code, Data, and Error Check — relative to the direction.)

ERROR CHECK	DATA	FUNCTION CODE	ADDRESS
INFORMATION USED BY RECEIVING DEVICE TO CHECK FOR ERRORS	RELATIVE ADDRESS OF REGISTER NUMBER	(03) READ HOLDING REGISTER	QUERY FOR SLAVE NUMBER 1

MODBUS MASTER

PC SLAVE NO

ADDRESS	FUNCTION CODE	DATA	ERROR CHECK
RESPONSE FROM SLAVE NUMBER 1	(03) READ HOLDING REGISTER	VALUE CONTAINED IN SPECIFIED HOLDING REGISTER	INFORMATION USED BY RECEIVING DEVICE TO CHECK FOR ERRORS

Figure 1-6. Simulated Query and Response Messages

SECTION 2
EXCEPTION RESPONSES

Programming or operation errors are those involving illegal data in a message, no response from PC to its interface unit, or difficulty in communicating with a slave. These errors result in an exception response from either the master computer software (Modbus Communications Handler) or the PC slave, depending on the type of error. The exception response codes are listed in Table 2-1. When a PC slave detects one of these errors, it sends a response message to the master consisting of slave address, function code, error code and error check fields. To indicate that the response is a notification of an error, the high order bit of the function code is set to 1. Figure 2-1 and 2-2 give an example of an incorrect query and the subsequent exception response codes.

Table 2-1. Exception Response Codes

CODE	NAME	MEANING
01	ILLEGAL FUNCTION	The message function received is not an allowable action for addressed slave. If a poll command was issued, indicates no program function preceded it.
02	ILLEGAL DATA ADDRESS	The address referenced in the data field is not an allowable address for the addressed slave location.
03	ILLEGAL DATA VALUE	The value referenced in the data field is not allowable in the addressed slave location.
04	FAILURE IN ASSOCIATED DEVICE	The slave's PC has failed to respond to a message or an abortive error occurred. (See Note 1.)
05	ACKNOWLEDGE	The slave PC has accepted and is processing the long duration program command. Issue a POLL PROGRAM COMPLETE message to find out when processing is finished. A poll message sent to the PC before it is finished will result in a rejected message response. (See Note 2.)
06	BUSY, REJECTED MESSAGE	The message was received without error, but the PC is engaged in processing a long duration program command. Retransmit later, when the PC may be free. (See Note 2.)

Table 2-1. Exception Response Codes (cont.)

CODE	NAME	MEANING
07	NAK-NEGATIVE ACKNOWLEDGMENT	The PROGRAM function just requested cannot be performed. Issue poll to obtain detailed device-dependent error information. Valid for PROGRAM/POLL 13 and 14 only.

NOTE 1: For function codes 1-19, an EXCPT Code 4 may indicate that only part of the associated query was processed before an irrecoverable error occurred within the responder station. Receipt of EXCPT Code 4 requires immediate supervisory notification.

NOTE 2: The 884 supports exception codes 5 and 6 only as device dependent error responses to function code 18. See Appendix A of the 384 Modbus Programming Protocol manual.

Refer to Appendix A of the specific device dependent manual for responses which can be obtained after exception response Codes 5, 6, and 7.

CODE	NAME	MEANING
08	MEMORY PARITY ERROR	The Modbus read of extended memory checks memory bits being accessed. Retries should be attempted as the error might not recur. If all retries fail, service may be required.

QUERY MESSAGE

SLAVE ADDR	FUNC	H.O. START ADDR	L.O. START ADDR	H.O. NO. OF COILS	L.O. NO. OF COILS	ERROR CHECK FIELD	
0A	01	04	A1	00	01	4F	LRC

Figure 2-1. Read Coil-Status Query Message

This query requests the status of Input 1245 in Slave No.10 and if this controller was a 1K-484 then this is an invalid input number. Consequently, the following error response might be generated:

SLAVE ADDR	FUNC	EXCEPTION CODE	ERROR CHECK	
0A	81	02	73	LRC

Figure 2-2. Read Coil Status Error Response

The function code field is the original function code with the high order bit set. Exception Code 02 indicated an illegal data address.

SECTION 3
DETAILED EXPLANATION OF MODBUS FUNCTIONS

The purpose of this section is to define the general format for the specific commands available to programmers of the Modbus System. The form of each query message, an example in ASCII transmission mode and an explanation of the function the query message performs is provided. Normal response messages are also included.

An overview description of the protocol and messages are described in Section 1.

Messages with function Codes 1-6, 15, & 16 indicate specifically which location(s) in the programmable controller is (are) to be accessed. Function Codes 1, 5, & 15 refer to coils (0XXX(X)). Code 2 refers to inputs (1XXX(X)), Code 4 refers to input registers (3XXX(X)), and Codes 3, 6, and 16 refer to holding registers (4XXX(X)). All address references in the Modbus messages are numbered relative to zero. For example the first holding register in a 584 would be register 40001 and would be referenced as 0000. Similarly, coil 00127 would be 0126 (decimal = 007E (hex). All numbers in a Modbus format are entered in hexadecimal.

The examples given in this section will present the protocol independent of RTU or ASCII framing considerations. See Figures 1-4 and 1-5 for ASCII & RTU framing examples. The programmer implementing the software may use the following method to correctly frame the protocol for their specific application.

The protocol will be presented as much as possible throughout this section in the format shown in Figure 3-1. Numbers represented are in hexadecimal.

ADDR	FUNC	DATA START REG HO	DATA START REG LO	DATA # OF REGS HO	DATA # OF REGS LO	ERROR CHECK FIELD	
06	03	00	6B	00	03	89	LRC

Figure 3-1. Presentation Example for Protocol

The example given is Read Output Registers 4108-4110 for the Modbus Slave Interface Unit addressed as 06. This message when specifically formatted either RTU or ASCII will look as follows:

QUERY MESSAGE		RTU		ASCII	
Header		None			Colon
Address		0000	0110	0	6
Function		0000	0011	0	3
Starting	H.O.	0000	0000	0	0
Register	L.O.	0110	1011	6	B
Quantity of	H.O.	0000	0000	0	0
Registers	L.O.	0000	0011	0	3
Error		0111	0101	8	9
Check		1010	0000		
Trailer		None		CR	LF
		8 Bytes Total		17 Bytes Total	

RESPONSE MESSAGE		RTU		ASCII	
Header		None			Colon
Address		0000	0110	0	6
Function		0000	0011	0	3
Byte Count		0000	0110	0	6
	H.O.	0000	0010	0	2
	L.O.	0010	1011	2	B
Data	H.O.	0000	0000	0	0
	H.O.	0000	0000	0	0
	L.O.	0110	0011	6	3
Error Check		CRC		6	1
Trailer		None		CR	LF
		11 Bytes Total		23 Bytes Total	

It will always be the case that the ASCII message in its final form will be approximately twice the length in bytes of the comparable RTU message.

The Normal Response message format has an implied or explicit message length depending on the Modbus function code. The explicit form is used for variable length Responses. The byte count in the response refers to the number of 8-bit (1 byte) data fields included in the response. In the RTU communication mode, a byte of information is transmitted as a single 8-bit character. In the ASCII communication mode, each 8-bit ASCII character contains 4 data bits and 4 bits for formatting the specific ASCII character sent., consequently two ASCII characters are required to transmit 8 bits (1 byte) of information. For example, address 06 is transmitted as '00000110' in the RTU mode, and '0110000' + '0110110' (ASCII 0 and 6) in the ASCII MODE. (data, data)

This general response format is used when responses are of a variable nature. For fixed length responses the byte count field is omitted and only the data fields are supplied. These functions are thought of as implied length functions. See Appendix C for implied length summaries. Exception Response message formats are discussed in Appendix A of the device specific manuals.

Table 3-1. Modbus Function Codes Supported By 184/384, 484, & 584 Controllers

Function Code	Description	184/384	484	584	884	Micro 84
1	Read Coil Status	Y	Y	Y	Y	Y
2	Read Input Status	Y	Y	Y	Y	Y
3	Read Holding Reg	Y	Y	Y	Y	Y
4	Read Input Reg.	Y	Y	Y	Y	Y
5	Force Coil	Y	Y	Y	Y	Y
6	Load Register	Y	Y	Y	Y	Y
7	Read Exception Status	Y	Y	Y	Y	Y

Table 3-1. Modbus Function Codes Supported By 184/384, 484, & 584 Controllers (cont.)

Function Code	Description	184/384	484	584	884	Micro 84
8	Loop Back Diagnostic	Y	Y	Y	Y	Y
9	Program 484	N	Y	N	N	N
10	Poll 484	N	Y	N	N	N
11	Comm. Event Counter	Y	N	Y	N	N
12	Comm. Event Log	Y	N	Y	N	N
13	Program-General	Y	N	Y	N	N
14	Poll-General	Y	N	Y	N	N
15	Force Multiple Coils	Y	Y	Y	Y	Y
16	Load Multiple Regs.	Y	Y	Y	Y	Y
17	Report Slave I.D.	Y	Y	Y	Y	Y
18	Program	N	N	N	Y	Y
19	Reset Communication Link	N	N	N	Y	Y
20	Read General Reference	N	N	Y	N	N
21	Write General Reference	N	N	Y	N	N

3.1 READ OUTPUT STATUS (FUNCTION CODE 01)

QUERY

This function allows the user to obtain the ON/OFF status of logic coils used to control discrete outputs from the addressed slave only. Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial coil address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 coils to be obtained at each request; however, the specific slave device may have restrictions that lower the maximum quantity (see Appendix B). The coils are numbered from zero; (coil number 1 = zero, coil number 2 = one, coil number 3 = two, etc.).

Figure 3-2 is a sample read output Status Request to read coils 0020 to 0056 from slave device number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD
11	01	00	13	00	25	B6 LRC

Figure 3-2. Read Output Status Query Message

RESPONSE

An example response to Read Output Status is as shown in Figure 3-3. The data is packed one bit for each coil. The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each coil (1 = ON, 0 = OFF). The low order bit of the first character contains the addressed coil, and the remainder follow. For coil quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as quantity of RTU characters, i.e., the number is the same whether RTU or ASCII is used.

Since the slave interface device is serviced at the end of a controller's scan, data will reflect coil status at the end of the scan. Some slaves will limit the quantity of coils provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status from sequential scans.

ADDR	FUNC	BYTE COUNT	DATA COIL STATUS 20-27	DATA COIL STATUS 28-35	DATA COIL STATUS 36-43	DATA COIL STATUS 44-51	DATA COIL STATUS 52-56	ERROR CHECK FIELD
11	01	05	CD	6B	B2	0E	1B	D6

Figure 3-3. Read Output Status Response Message

The status of coils 20-27 is shown as CD(HEX) = 1100 1101 (Binary). Reading left to right, this shows that coils 27, 26, 23, 22, and 20 are all on. The other coil data bytes are decoded similarly. Due to the quantity of coil statuses requested, the last data field, which is shown as 1B (HEX) = 0001 1011 (Binary), contains the status of only 5 coils (52-56) instead of 8 coils. The 3 left-most bits are provided as zeros to fill the 8-bit format.

3.2 READ INPUT STATUS (FUNCTION CODE 02)

QUERY

This function allows the user to obtain the ON/OFF status of discrete inputs in the addressed slave PC Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial input address to be read (Starting Address) and the number of locations that will be interrogated to obtain status data.

The addressing allows up to 2000 inputs to be obtained at each request; however, the specific slave device may have restrictions that lower the maximum quantity (see Appendix B). The inputs are numbered from zero; (input 10001 = zero, input 10002 = one, input 10003 = two, etc., for a 584).

Figure 3-4 is a sample read input Status Request to read inputs 10197 to 10218 from 584 slave number 17.

ADDR	FUNC	DATA START PT HO	DATA START PT LO	DATA # OF PTS HO	DATA # OF PTS LO	ERROR CHECK FIELD	
11	02	00	C4	00	16	13	LRC

Figure 3-4. Read Input Status Query Message

RESPONSE

An example response to Read Input Status is as shown in Figure 3-5. The data is packed one bit for each input.

The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data will be packed with one bit for each input (1 = ON, 0 = OFF). The lower order bit of the first character contains the addressed input, and the remainder follow. For input quantities that are not even multiples of eight, the last characters will be filled in with zeros at high order end. The quantity of data characters is always specified as a quantity of RTU characters, i.e., the number is the same whether RTU or ASCII is used.

Since the slave interface device is serviced at the end of a controller's scan, data will reflect input status at the end of the scan. Some slaves will limit the quantity of inputs provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status from sequential scans.

ADDR	FUNC	BYTE COUNT	DATA DISCRETE INPUT 10197-10204	DATA DISCRETE INPUT 10205-10212	DATA DISCRETE INPUT 10213-10218	ERROR CHECK FIELD	
11	02	03	AC	DB	35	2E	LRC

Figure 3-5. Read Input Status Response Message

The status of inputs 10197-10204 is shown as AC (HEX) = 1010 1100 (Binary). Reading left to right, this shows that inputs 10204, 10202, 10200, and 10199 are all on. The other input data bytes are decoded similarly.

Due to the quantity of input statuses requested, the last data field which is shown as 35 HEX = 0011-0101 (Binary) contains the status of only 6 inputs (10213-10218) instead of 8 inputs. The two left-most bits are provided as zeros to fill the 8-bit format.

3.3 READ OUTPUT REGISTERS (FUNCTION CODE 03)

QUERY

Read output Registers (03) allows the user to obtain the binary contents of holding registers in the addressed slave.

These registers can store the numerical values of associated timers and counters which can be driven to external devices.

The addressing allows up to 125 registers to be obtained at each request; however, the specific slave device may have restrictions that lower this maximum quantity (See Appendix D). The registers are numbered from zero (40001 = zero, 40002 = one, etc.).

Broadcast mode is not allowed.

The below example reads registers 40108 through 40110 from slave 584 number 17.

ADDR	FUNC	DATA START REG HO	DATA START REG LO	DATA # OF REGS HO	DATA # OF REGS LO	ERROR CHECK FIELD	
11	03	00	6B	00	03	7E	LRC

Figure 3-6. Read Output Register Query Message

RESPONSE
The addressed slave responds with its address and the function code, followed by the information field. The information field contains 2 bytes describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are two bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Since the slave interface device is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Some slaves will limit the quantity of register content provided each scan; thus for large register quantities, multiple transmissions will be made using register content from sequential scans.

In the below example the registers 40108-40110 have the decimal contents 555, 0, and 100 respectively.

ADDR	FUNC	BYTE COUNT	DATA OUTPUT REG H.O. 40108	DATA OUTPUT REG L.O. 40108	DATA OUTPUT REG H.O. 40109	DATA OUTPUT REG L.O. 40109	DATA OUTPUT REG H.O. 40110	DATA OUTPUT REG L.O. 40110	ERROR CHECK FIELD
11	03	06	02	2B	00	00	00	64	55

Figure 3-7. Read Output Register Response Message

3.4 READ INPUT REGISTERS (FUNCTION CODE 04)

QUERY
Function Code 04 obtains the contents of the controller's input registers. These locations receive their values from devices connected to the I/O structure and can only be referenced, not altered from within the controller or via the Modbus. The addressing allows up to 125 registers to be obtained at each request; however, the specific slave device may have restrictions that lower this maximum quantity (See Appendix B). The registers are numbered from zero (30001 = zero, 30002 = one, etc.). Broadcast mode is not allowed.

The below example requests the contents of register 3009 in 484 slave number 17. In the response the contents is decimal value 1337.

ADDR	FUNC	DATA START REG HO	DATA START REG LO	DATA # OF REGS HO	DATA # OF REGS LO	ERROR CHECK FIELD	
11	04	00	08	00	01	E2	LRC

Figure 3-8. Read Input Register Query Message

RESPONSE
The addressed slave responds with its address and the function code followed by the information field. The information field contains 2 bytes describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are 2 bytes each, with the binary content right justified within each pair of characters. The first byte includes the high order bits and the second, the low order bits.

Since the slave interface is normally serviced at the end of the controller's scan, the data will reflect the register content at the end of the scan. Each PC will limit the quantity of register contents provided each scan; thus for large register quantities, multiple PC scans will be required, and the data provided will be from sequential scans.

In the below example the register 3009 contains the decimal value 0.

ADDR	FUNC	BYTE COUNT	DATA INPUT REG HO 3009	DATA INPUT REG LO 3009	ERROR CHECK FIELD	
11	04	02	00	00	E9	LRC

Figure 3-9. Read Input Register Response Message

3.5 FORCE SINGLE COIL (FUNCTION CODE 05)

CAUTION

COMMAND (05) WILL OVERRIDE BOTH PC MEMORY PROTECT AND COIL DISABLE STATE

QUERY
This message forces a single coil either ON or OFF. Any coil that exists within the controller can be forced to either state (ON or OFF). However, since the controller is actively scanning, unless the coil is disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 0001 = zero, coil 0002 = one, etc.). The data value 65,280 (FF00 HEX) will set the coil ON and the value zero will turn it OFF; all other values are illegal and will not effect that coil.

The use of slave address 00 (Broadcast Mode) will force all attached slaves to modify the desired coil.

NOTE

Functions 5, 6, 15 and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

The below example is a request to slave number 17 to turn ON coil 0173.

ADDR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON OFF IND	DATA	ERROR CHECK FIELD	
11	05	00	AC	FF	00	3F	LRC

Figure 3-10. Force Single Coil Query Message

The normal response to the Command Request is to retransmit the message as received after the coil state has been altered.

ADDR	FUNC	DATA COIL # HO	DATA COIL # LO	DATA ON OFF	DATA	ERROR CHECK FIELD	
11	05	00	AC	FF	00	3F	LRC

Figure 3-11. Force Single Coil Response Message

The forcing of a coil via Modbus function 5 will be accomplished regardless of whether the addressed coil is disabled or not.

NOTE

The Modbus protocol does not include standard functions for testing or changing the DISABLE state of discrete inputs or outputs. Where applicable, this may be accomplished via device specific Program commands. See function Codes 9 and 13.

One additional caution: Coils that are unprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function Code 5 and (even months later), an output is connected to that coil, the output will be "hot".

3.6 PRESET SINGLE REGISTER (FUNCTION CODE 06)

CAUTION

Function (06) will override controller memory protect.

QUERY

Function (06) allows the user to modify the contents of a holding register. Any holding register that exists within the controller can have its contents changed by this message. However, since the controller is actively scanning, it also can alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller (10-bit for 484 and 16-bit for 184/384 and 584); unused high order bits must be set to zero. When used with slave address zero (Broadcast mode) all slave controllers will load the specified register with the contents specified.

NOTE

Functions 5, 6, 15 and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

ADDR	FUNC	DATA REG # HO	DATA REG # LO	DATA VALUE HO	DATA VALUE LO	ERROR CHECK FIELD	
	06	00	87	03	9E	C1	LRC

Figure 3-12. Preset Single Register Query Message

RESPONSE

The normal response to a preset single register request is to retransmit the qu message after the register has been altered.

ADDR	FUNC	DATA REG 4136 HO	DATA REG 4136 LO	DATA	DATA	ERROR CHECK FIELD	
11	06	00	87	03	9E	C1	LRC

Figure 3-13. Preset Single Register Response Message

3.7 READ EXCEPTION STATUS (FUNCTION CODE 07)

QUERY

In many cases a short message requesting the status of certain events in a controller is desired. The read exception status code is designed to provide this functionality.

Function Code 7 allows the user to interrogate the status of eight coils within the controller. These coils can be programmed to hold information of the PC's control situation (e.g., machine ON/OFF, heads retracted, safeties satisfied, receipt in process error conditions exist, etc.). Slave address 00 (Broadcast Mode) is not supported with the exception Status Request Message.

The eight coil statuses returned by this command depend on the slave PC type. The status of coils 1-8 are returned if the slave PC is a 584, 184/384, or Micro 84 and the status of coils 257-264 are returned if the slave PC is a 484. If the slave PC is an 884, the states of coils 761-768 are returned. See Table 3-2 for special coil assignments.

Table 3-2. Exception Status Special Coil Assignments

Controller	Coil No.	Assignment
484	257	Battery Status
884	761	Battery Status
884	762	Memory Protect Status
884	763	Remote I/O Health Status

The below example displays a request to slave number 17 to respond with exception status.

ADDR	FUNC	ERROR CHECK FIELD	
11	07*	E8	LRC

Figure 3-14. Read Exception Status Query Message

*No data fields required for this function.

RESPONSE

The normal response contains the status of the eight coils, packed into one data byte, one bit per coil.

SLAVE ADDR	FUNC	COIL DATA	ERROR CHECK FIELD	
11	07	6D	7B	LRC

Figure 3-15. Read Exception Status Response Message.

In this example, if slave 17 is a 484 controller, since 6D (HEX) = 0110 1101 (Binary), coil 257 is ON (batteries OK), coils 259, 260, 262, and 263 are ON and coils 258, 261, and 264 are OFF. The user determines what the last 7 coils are programmed to reflect.

In the 184/384 and 584 function Code 7 reads coils 1-8. The 184/384 PC line uses core memory, therefore battery back-up for memory retention is not required and all eight coils are available for the user program. The 584 has a user configurable battery OK coil. The user may configure it as any of coils 1-8 such that it is read by this command.

3.8 LOOPBACK TEST (FUNCTION CODE 8)

QUERY

The purpose of the Loopback Test is to test the communications system: it does not affect the content of the controller.

Variations in the response may indicate faults in the Modbus system. The information field contains 2 bytes for the designation of diagnostic code followed by 2 bytes to designate the action to be taken.

The use of the Loopback Test also allows the user to fetch the Diagnostic Register contents, a valuable tool for testing system operation. In addition, the use of diagnostic Codes 11-13 enable the user to interrogate the Modbus Slave's Bus message, Bus CRC error and Bus Exception error counters. These counters supply information that is useful for communications error analysis.

The next four diagnostic Codes (14-17) enable the user to obtain counter information from the slave associated controller.

These counters accumulate data on the number of messages received and acted upon; the number of messages requiring no response; the number of issued not acknowledge responses: and, number of times the device was busy when a program command was issued. The last codes (18-20) associated with loopback Test are reserved for Gould use.

An example of a Loopback Test given below requests a simple return of the query message (Diagnostic Code 0) sent to slave number 17.

ADDR	FUNC	DATA DIAG CODE HO	DATA DIAG CODE LO	DATA*	DATA*	ERROR CHECK FIELD	
11	08	00	00	A5	37	0B	LRC

*These are considered to be the information fields for diagnostic mode.

Figure 3-16. Loopback Test Return Query Data Query

RESPONSE

ADDR	FUNC	DATA DIAG CODE HO	DATA DIAG CODE LO	DATA	DATA	ERROR CHECK FIELD	
11	08	00	00	A5	37	0B	LRC

Figure 3-17. Loopback Test Return Query Data Response

DIAGNOSTIC CODES (SEE TABLE 3-3 FOR CODES SUPPORTED FOR EACH PC)

HEX	DECIMAL		
00	0	=	Return Query Data
01	1	=	Restart Comm Option (no response)
02	2	=	Return Diagnostic Register
03	3	=	Change Input Delimiter Character
04	4	=	Force Slave to Listen Only Mode
0A	10	=	Clear Counters and Diagnostic Register
0B	11	=	Return Bus Message Count
0C	12	=	Return Bus CRC Error Count
0D	13	=	Return Bus Exception Error Count
0E	14	=	Return Slave Message Count
0F	15	=	Return Slave No Response Count
10	16	=	Return Slave NAK Count
11	17	=	Return Slave Busy Count
12	18	=	Return Bus Character Overrun Count
13	19	=	Return Overrun Error Count
14	20	=	Clear Overrun Error Count and Flag

NOTE

Codes 5-9 and 21-65,535 are illegal codes.

Table 3-3. Supported Diagnostic Codes for the Loopback Test Modbus Function

DIAG CODE	DEFINITION	184/384	484	584	884	MICRO 84
00	Return Query Data	Y	Y	Y	Y	Y
01	Restart Comm Option (No response from slave com. option Error = 199) 00/FF = Halt/Continue on Error	Y	Y	Y	Y	Y
02	Return Diagnostic Register	Y	Y	Y	Y	Y
03	Change ASCII Input Delimiter	Y	Y	Y	N	N
04	Force Listen Only Mode.	Y	Y	Y	Y	Y
05-09	RESERVED	Y	Y	Y	N	N
10	Clear Counters and Diagnostic Register	Y	Y	CLEARS CTRS ONLY	N	N
11	Return Bus Message Count	Y	Y	Y	N	N
12	Return Bus Comm. Error Count	Y	Y	Y	N	N
13	Return Bus EXCP Error Count	Y	Y	Y	N	N
14	Return Slave Message Count	Y	Y	Y	N	N
15	Return Slave No Response Count	Y	Y	Y	N	N
16	Return Slave NAK Count	Y	Y	Y	N	N
17	Return Slave Busy Count	Y	Y	Y	N	N
18	Return Bus Char. Overrun Count	Y	Y	Y	N	N
19	Return Overrun Error Count	N	N	N	Y	N
20	Clear Overrun Error Count & Flag	N	N	N	Y	N
21-65, 535	RESERVED	NA	NA	NA	NA	NA

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	00	ANY	DATA

RETURN QUERY DATA
The data passed in the information field will be returned to the master via the addressed Modbus slave. The entire message returned should be identical to the message transmitted by the master, field-per-field.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	01	00	00
		FF	00

RESTART COMM. OPTION
This code causes the peripheral port to be reset and all counters to be cleared. If "CONTINUE ON ERROR" (FF00) is specified, the COMM EVENT LOG will be cleared. A normal response is sent before the restart is executed. If the port is in Listen Only Mode when the Restart command is received, no response will be initiated. Restart is the only command that will bring the port out of Listen only Mode. If this command is received, a restart is attempted, executing the power-up confidence tests. Successful completion of these tests will put the unit back in on-line mode, with the cause of entry into LCM saved in the Diagnostic Register.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	02	00	00

RETURN DIAGNOSTIC REGISTER
This code returns a 16-bit diagnostic register. See Tables 3-2, 3-3, 3-4, 3-4A & 3-4B for assignment of bits in the diagnostic register for each type of slave controller.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	03	DATA CHAR	00

CHANGE INPUT DELIMITER CHAR.
The character passed in the information field provides a means to alter [illegible] character as the delimiter character in an ASCII message.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	04	00	00

FORCE SLAVE TO LISTEN ONLY MODE (LOM)
No response is issued. When LOM is entered, all active controls are immediately turned off and the Ready watchdog timer is allowed to expire, locking these controls off. This isolates a failed unit from other stations on the line, allowing the others to continue full communication. It also prevents a failed communications port from interfering with the local application.

While in LOM the Modbus received data is monitored (but no responses are transmitted). The only query that will be recognized and processed while in LOM is a maintenance restart command (function Code 8, diagnostic Code 1).

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	0A	00	00

CLEAR COUNTERS AND DIAGNOSTIC REGISTER
All counters and the diagnostic register are cleared to zero from their present value except in a 584 where the diagnostic register is not cleared. Counters are also cleared by power up.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	0B	00	00

RETURN BUS MESSAGE COUNT
The information field returns with the number of messages that the addressed Modbus slave interface has processed since the last restart was issued (or power up).

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	0C	00	00

RETURN BUS CRC ERROR COUNT
The information field returns with the number of CRC errors encountered for the addressed Modbus slave interface unit since the last restart or power up was issued.

DETAILED EXPLANATION OF MODBUS FUNCTIONS

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	0D	00	00

RETURN BUS EXCEPTION ERROR COUNT
The information field returns to the master with the number of exception codes returned by the addressed Modbus slave interface unit since the last restart of power up was issued.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
J0	0E	00	00

RETURN SLAVE MESSAGE COUNT
The information field returns to the master with the number of messages which were addressed to the PC since the last restart or power up was issued.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	0F	00	00

RETURN SLAVE NO RESPONSE COUNT
Returns in the information field the number of times the attached PC has failed to respond to the interface unit since the addressed Modbus slave last restart or power up was issued.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	10	00	00

RETURN SLAVE NAK COUNT
Returns in the information field the number of times the attached PC responded to the addressed Modbus slave interface unit with a NAK (Negative Acknowledgment) since the last restart. clear counters, or power up was issued.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	11	00	00

RETURN SLAVE BUSY COUNT
Returns in the information field the number of times the attached PC responded to the addressed Modbus slave interface unit as busy since the last restart, clear counters, or power up was issued.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	12	00	00

RETURN BUS CHARACTER OVERRUN COUNT
Returns in the information field the number of times a message was not handled due to characters arriving at the UART faster than they could be stored or loss of a character due to a hardware malfunction.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	13	00	00

RETURN OVERRUN ERROR COUNT
Returns in the information field the number of times a Modbus message was not handled at the Modbus IOP due to a character overrun error.

DIAG CODE FUNCTION H.O.	L.O.	INFORMATION FIELD H.O.	L.O.
00	14	00	00

CLEAR OVERRUN ERROR COUNT AND FLAG
Clears the overrun error count and the flag which is set when the error count is non-zero. The state of the error flag is shown in bit o of the diagnostic register of the 884 (see Table 3-7A).

Table 3-4. Diagnostic Register Bit Assignments for the 184/384 Controller

The following information is contained in the diagnostic register (bit 15 is the high order bit and the description indicates the meaning when the bit is):

BIT	DESCRIPTION
0	Continue On Error
1	Run Light Failed
2	T-Bus Test Failed
3	Asynchronous Bus Test Failed
4	Force Listen Only Mode
5	Spare
6	Spare
7	ROM Chip number 0 Test Failed
8	Continuous ROM Checksum Test In Execution
9	ROM Chip number 1 Test Failed
10	ROM Chip number 2 Test Failed
11	ROM Chip number 3 Test Failed
12	RAM Chip 5000-53FF Test Failed
13	RAM Chip 6000-67FF Test Failed (Even Addresses)
14	RAM Chip 6000-67FF Test Failed (Odd Addresses)
15	Timer Chip Test Failed

Table 3-5. Diagnostic Register Bit Assignments for the 484 Controller

The following information is contained in the diagnostic register (Bit 15 is the high order bit and the description indicates the meaning when the bit is set):

BIT	DESCRIPTION
0	Continue on error
1	CPU test or run light failed
2	Parallel port test failed
3	Asynchronous bus test failed

Table 3-5. Diagnostic Register Bit Assignments for the 484 Controller (cont.)

BIT	DESCRIPTION
4	Timer 0 test failed
5	Timer 1 test failed
6	Timer 2 test failed
7	ROM chip 0000-07FF test failed
8	Continuous ROM checksum test in execution
9	ROM chip 0800-0FFF test failed
10	ROM chip 1000-17FF test failed
11	ROM chip 1800-1FFF test failed
12	RAM chip 400-40FF test failed
13	RAM chip 4100-41FF test failed
14	RAM chip 4200-42FF test failed
15	RAM chip 4300-43FF test failed

Table 3-6. Diagnostic Register Bit Assignments for the 584 Controller

The following information is contained in the diagnostic register (Bit 15 is the high order bit and the description indicates the meaning when the bit is set):

BIT	DESCRIPTION
0	Illegal configuration
1	Backup checksum error (in high-speed RAM)
2	Logic checksum error
3	Invalid node type
4	Invalid traffic cop type
5	CPU/Solve diagnostic failure
6	Real time clock failure
7	Watchdog timer failure (scan time exceeded 250 milliseconds)
8	No end of logic node detected, or number of end of segment words (DOIO) does not match number segments configured

Table 3-6. Diagnostic Register Bit Assignments for the 584 Controller (cont.)

BIT	DESCRIPTION
9	State RAM test failed
10	Start of Network (SON) did not begin segment
11	Bad order of solve table
12	Illegal peripheral intervention
13	Dim awareness flag
14	Unused
15	Peripheral Port "STOP" (see NOTE)

Table 3-7. Diagnostic Register Bit Assignments For the 884 Controller

The following information is contained in the diagnostic register (bit 15 is the high order bit and the description indicates the meaning when the bit is set):

Bit	Description
15-9	Reserved for future options failure information
8	Set if PC SCAN TASK has exceeded its time slice (e.g., too much user logic).
7	Set if at least one table RAM checksum failed
6	Main CPU failed
5	Remote IO failure
4	Ourbus IOP failure
3	Modbus option failure
2	Modbus IOP failure
1	Modbus option overrun errors flag
0	Modbus IOP overrun errors flag

NOTE Denotes "SOFT" stop, which was induced in a controlled manner. All other STOP's are classified as errors.

3.9 FETCH COMMUNICATIONS EVENT COUNTER (FUNCTION CODE 11)

QUERY

Fetch Event Counter returns a 2 byte status word and a 2 byte event counter.

The event counter, controlled by the PC, gets incremented once for every successful message completion (it will not be incremented for exception responses, poll or fetch event counter commands). The intent is to provide a means whereby a master can issue a single query and subsequently determine whether the operation was successfully performed, especially when a communication error occurred on that command or its response.

The event counter may be reset by issuing a loopback function (8) diagnostic Code 10 (clear counters) or a diagnostic Code 1 (restart comm with halt option).

The status word will be set to all ones if a previously issued program command is still being processed (a busy condition). Otherwise, the status word is set to zero.

ADDR	FUNC	ERROR CHECK FIELD	
11	0B	E4	LRC

Figure 3-18. Fetch Event Counter Query Message

RESPONSE

In this example response the status word indicates a program function is still in progress and 264 (108 Hex) events have been counted by the controller.

ADDR	FUNC	HO STATUS	LO STATUS	HO EVT CNT	LO EVT CNT	ERROR CHECK FIELD	
11	0B	FF	FF	01	08	DD	LRC

Figure 3-19. Fetch Event Counter Response Message

NOTE

This command is not supported by the following model PC's: Micro 84, 484 and 884.

3.10 FETCH COMMUNICATIONS EVENT LOG (FUNCTION CODE 12)

QUERY

Fetch Communication Event log returns a 2 byte status word, a 2 byte event counter, a 2-byte message counter, and 64 event bytes.

The status word and event byte are identical to that returned in function Fetch Event Counter (11).

The message counter returns the number of messages which the controller processed since the last restart with halt option or power up. It is identical to data returned in function Loopback diagnostic code bus message counter (Function 8, diagnostic Code 11).

The event bytes are a circular array, one byte of information corresponding to each Bus send or receive operation (and certain internal operations) for that controller. The status bytes are entered in chronological order. Any wrap-around will always overwrite the oldest status byte with the current one. The maximum size of the communication event array is 64 bytes.

Event bytes may be any of 4 types:

A) Slave Bus Receive: byte is stored when query is received (before processing).

Bit 0 Reserved
Bit 1 - Set if Comm Error
Bit 2 - Reserved
Bit 3 - Reserved
Bit 4 - Set if Character Overrun
Bit 5 - Set if in Listen Only Mode
Bit 6 - Set if Broadcast
Bit 7 1

B) Slave Bus Send: byte is stored when processing and/or response is finished if one was started.

Bit 0 Set if Read Exception sent (Exception Code 1-3)
Bit 1 Set if Slave Abort Exception sent (Exception Code 4)
Bit 2 - Set if Slave Busy Exception sent (Exception Code 5-6)
Bit 3 - Set if Slave Program NAK Exception sent (Exception Code 7)
Bit 4 - Set if Write Timeout Error has occurred.
Bit 5 Set if in Listen Only Mode
Bit 6 - 1
Bit 7 0

C) Entered Listen Only Mode: Byte is stored whenever Listen Only Mode is entered.

Bit 0 - 0
Bit 1 - 0
Bit 2 - 1
Bit 3 - 0
Bit 4 - 0
Bit 5 - 0
Bit 6 - 0
Bit 7 - 0

HEX Value = 20

D) Initiated Communication Restart: one byte is stored if "Continue on Error Mode". Entire log is stored if "Stop on Error" (i.e., log is cleared to all zeros).

Bit 0 - 0
Bit 1 0
Bit 2 - 0
Bit 3 - 0
Bit 4 0
Bit 5 - 0
Bit 6 - 0
Bit 7 0

ADDR	FUNC	ERROR CHECK FIELD
11	0B	E4	LRC

Figure 3-20. Fetch Communications Event Log Query Message

RESPONSE

In this example response, status word indicates no event is in progress. 264 (108 HEX) events have been counted by the controller. The message counter is set to 288 (120 HEX). The most recent event byte is "Entered Listen Only Mode" type (20). The second most recent byte is a "Com Restart" type (00).

ADDR	FUNC	BYTE CNT	HO STATUS	LO STATUS	HO EVT CNT	LO EVT CNT	HO MSG CNT	LO MSG CNT	N DATA	N-1 DATA	ERROR CHECK FIELD
11	0C	46	00	00	01	08	01	20	20	00	53

Figure 3-21. Fetch Communications Event Log Response Message

NOTE

This Modbus command is not supported by Gould PC model types 484, 884, and Micro-84.

3.11 FORCE MULTIPLE COILS (FUNCTION CODE 15)

CAUTION

Command (15) will override both PC memory protect and coil disable.

QUERY

This message forces each coil in a consecutive block of coils to a desired ON or OFF state. Any coil that exists within the controller can be forced to either state (ON or OFF). However, since the controller is actively scanning, unless the coils are disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 00001 = zero, coil 00002 = one, etc.). The desired status of each coil is packed in the data field, one bit for each coil (1 = ON, 0 = OFF).

The use of slave address 00 (Broadcast Mode) will force all attached slaves to modify the desired coils.

NOTE

Functions 5, 6, 15 and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

The following example forces 10 coils starting at address 20 (13 HEX). The [illegible] data fields, CD = 1100 and 00 = 0000 0000, indicate that coils 27, 26, 23, 22 a[illegible] are to be forced on.

ADDR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		BYTE CNT	DATA COIL STUS 20-27	DATA COIL STUS 28-29	ERROR CHECK FIELD	
11	0F	00	13	00	0A	02	CD	00	F4	LRC

Figure 3-22. Force Multiple Coils Query Message

RESPONSE

The normal response will be an echo of the slave address, function code, starting address, and quantity of coils forced.

ADDR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY			ERROR CHECK FIELD	
11	0F	00	13	00	0A		C3	LRC

Figure 3-23. Force Multiple Coils Response Message

The writing of coils via Modbus functions 15 will be accomplished regardless of whether the addressed coils are disabled or not.

NOTE

The Modbus protocol does not include standard functions for testing or changing the DISABLE state of discrete inputs or outputs. Where applicable, this may be accomplished via device specific program commands. See Function Code 13.

One additional caution: Coils that are unprogrammed in the controller logic program are not automatically cleared upon power up. Thus, if such a coil is set ON by function Code 15 and (even months later) an output is connected to that coil, the output will be hot.

3.12 PRESET MULTIPLE REGISTERS (FUNCTION CODE 16)

QUERY

CAUTION

Function (16) will override controller memory protect.

Holding registers existing within the controller can have their contents changed by this message (a maximum of 60 registers). However, since the controller is actively scanning, it also can alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller (16-bit for the 184/384 and 584); unused high order bits must be set to zero. When used with slave address zero (Broadcast Mode) all slave controllers will load the specified registers with the contents specified.

NOTE

Function Codes 5, 6, 15 and 16 are the only messages (other than Loopback Diagnostic Test) that will be recognized as valid for broadcast.

ADDR	FUNC	H.O. ADDR	L.C ADDR	QUANTITY		BYTE CNT	H.O. DATA	L.O. DATA	H.O. DATA	L.O. DATA	ERROR CHECK FIELD	
11	10	00	87	00	02	04	00	0A	01	02	45	LRC

Figure 3-24. Preset Multiple Coils Query Message

RESPONSE

The normal response to a function 16 query is to echo the address, function code, starting address and number of registers to be loaded.

ADDR	FUNC	H.O. ADDR	L.O. ADDR	QUANTITY		ERROR CHECK FIELD	
11	10	00	87	00	02	56	LRC

Figure 3-25 Preset Multiple Registers Response Message

3.13 REPORT SLAVE ID (FUNCTION CODE 17)

QUERY

Report Slave ID permits the user to obtain slave type, slave Run light and supplementary device dependent status and configuration information.

Slave ID = 0 for Micro 84

1 for 484

2 for 184, 384, 384A, 384B

3 for 584

8 for 884

Runlight = 00 for Run light off

Runlight = FF for Run light on

ADDR	FUNC	ERROR CHECK FIELD	
11	11	DE	LRC

Figure 3-26. Report Slave ID Query Message

RESPONSE

The general form of the response is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
					////	

Figure 3-27. Report Slave ID Response Message

3.13.1 184/384 SUPPORT OF FUNCTION CODE 17

An example response for a 184/384 controller with a Run light off is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
11	11	4A	02	00	////	

Figure 3-28. Report Slave ID Response Message For 184/384

The byte count contains the number of data bytes returned to the master. If the position of the switches on the J347 matches the actual controller type and the PIB table is normal, the byte count will be 4A (HEX). If the switches do not match the controller type only 4 bytes of data will be returned. These 4 bytes are:

1 BYTE Slave ID = 2 for 184/384

1 BYTE Run light status (0 = off, FF = on)

2 BYTES Status Word

BIT 0 = 0

BIT 1 = Status of Memory Protect (0 = off, 1 = on)

BIT 2 CONTROLLER TYPE For 184, BIT 2 = 0 and BIT 3 = 0
For 384, BIT 2 = 1 and BIT 3 = 0

BIT 3

BITS 4-5 = unused

In addition to these 4 bytes of data, if the switches on the J347 match the controller type and the PIB table is normal, 70 additional bytes of information will be returned. These bytes are transmitted in the following order:

2 BYTES PIB table starting address

2 BYTES Controller serial number

2 BYTES Executive I.D.

64 BYTES PIB table (See Table Below)

NOTE

If the 184/384 controller is stopped. these 70 bytes cannot be considered accurate information. The controller should be running for valid status information.

PIB TABLE

WORD
1 - Maximum Number of Output Coils
2 - Output Coil Enable Table
3 - Address of Input Coil/Run Table
4 - Number of Input Coils
5 - Input Coil Enable Table
6 - First Latch No. (Must be multiple of 16)
7 - Last Latch No. (Must be multiple of 16)
8 - Address of Input Registers
9 - Number of Input Registers
10 - Number of Output & Holding Registers
11 - Address of User Logic
12 - Address of Output Coil RAM Table
13 - Function Inhibit Mask
14 - Address of Extended Function Routine
15 - Address of Data Transfer Routine
16 - Address of Traffic Cop
17 - Spare
18 - Function Inhibit Mask
19 - Address of 'A' Mode History Table
20 - Request Table For DX Printer
21 - Number of Sequence Groups
22 - Address of Sequence Image Table
23 - Address of Sequence RAM
24 - Number of 50XX Registers
25 - Address of 50XX Table
26 - Address of Output Coil RAM Image
27 - Address of Input RAM Image
28 - Delayed Output Start Group
29 - Delayed Output End Group
30 - Watch Dog Line
31 - RAM Address of Latches
32 - Number of Delayed Outputs Groups

3.13.2 484 SUPPORT OF FUNCTION CODE 17

An example response for a 484 controller with a Run light on is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
11	11	05	03	FF	////	

Figure 3-29 Report Slave ID Response Message for 484

The five bytes of data for a 484 follow:

1 BYTE Slave ID = 1 for 484

1 BYTE Runlight Status (0 = off, FF = on)

3 BYTES Device Dependent (Refer to the 484 Database, PI-MBUS-302):

1 byte = System State
1 byte = First Configuration Byte
1 byte = Second Configuration Byte

3.13.3 584 SUPPORT OF FUNCTION CODE 17

An example response for a 584 controller with a Run light on is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
11	11	09	03	FF	////	

Figure 3-30. Report Slave ID Response Message for 584

The 9 bytes of data for a 584 are as follows:

1 BYTE Slave ID = 3 for 584

1 BYTE Runlight status (0 off, FF = on)

1 BYTE Number of 4K sections of page 0 memory

1 BYTE Number of 1K sections of State RAM

1 BYTE Number of segments in the 584

2 BYTES Machine State bits (word 65 (HEX) of 584 configuration table) — See the 584 Database (PI-MBUS-303)

2 BYTES Machine Stop Code (word 69 (HEX) of 584 configuration table) — See the 584 Database (PI-MBUS-303)

3.13.4 Micro 84 Support of Function Code 17

An example response for a Micro 84 controller with a runlight on is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
11	11	08	00	FF	////	

Figure 3-31. Report Slave I.D. Response Message for Micro 84

The 8 bytes of data for a Micro 84 are as follows:

1 BYTE	Slave ID = 0 for Micro 84
1 BYTE	Runlight Status (0 = off, FF = on)
1 BYTE	Current Port No. (will be 0 at present)
1 BYTE	Memory Size of Micro 84 (1 = 1K, 2 = 2K)
4 BYTES	Presently All Zeros

3.13.5 884 Support of Function Code 17

An example response for an 884 controller with a runlight on is as follows:

SLAVE ADDR	FUNC	BYTE COUNT	SLAVE ID	RUN LIGHT	DEVICE DEPENDENT	ERROR CHECK FIELD
11	11	08	08	FF	////	

Figure 3-32. Report Slave I.D. Response Message for 884

The eight bytes of data for the 884 are as follows:

1 BYTE		Slave I.D. (884 = 08)
1 BYTE		Runlight Status (0 = OFF, FF = ON)
1 BYTE		Current Port No. (1 = port 1 on CPU, 2 = port 2 (optic
1 BYTE		884 Application Memory Size (KBytes) (User Logic & State RAM) 2 Bytes = 1 Word
1 BYTE		Hook Bits
	bits 0-7	Reserved
1BYTE		Hook Bits
	bits 0-2	Reserved
	bit 3	Mapper Bypass — "1" = DO NOT EXECUTE STANDARD MAPPER
	bit 4	End of Scan Tests — "1" = TEST END OF SCAN HOOKS
	bit 5	Reserved
	bit 6	Logic Solver Bypass — "1" = DO NOT EXECUTE STANDARD LOGIC SOLVER

bit 7	Reserved
1 BYTE	Hook Bits
bits 0-7	Reserved
1 BYTE	Hook Bits
bits 0-7	Reserved

3.14 READ GENERAL REFERENCE (FUNCTION CODE 20)

QUERY

Read general reference query message requests information contained in Gould 584L extended memory files. Several sub-requests can be included in one message. Each sub-request reads a contiguous group of registers.

			SUB-REQUEST				SUB-REQUEST				
ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REQ ADDR	REG CNT	REF TYPE	FILE NO	REQ ADDR	REG CNT	ERROR CHECK FIELD
B	B	B	B	W	W	W	B	W	W	W	W(CRC) OR B(LRC)

Figure 3-33. Read General Reference Query Message Format.

			SUB-REQUEST				SUB-REQUEST				
ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REQ ADDR	REG CNT	REF TYPE	FILE NO	REQ ADDR	REG CNT	ERROR CHECK FIELD
11	14	0E	06	0002	0001	0005	06	0002	0009	0005	CRC OR LRC

Figure 3-34. SAMPLE: Read General Reference Query Message.

Byte Count = the total number of bytes in the read general reference message. excluding the address. function code. byte count. and the error check fields: that is all occurences of the following: reference type. file number. register address. register count. In the read query message illustrated in Figure 3-33 the byte count is 14. (0E in hexadecimal) because two sub-requests are being sent. each of which is 7 bytes long.

Reference Type = must be 6 (06 hexadecimal). 584L Extended Memory Storage Registers.

File number = the number of the file to be read. Extended memory is addressed as a group of files. each file containing up to 10.000 registers. File numbers can range from 1 to 10 depending upon the extended memory size added.

Starting Register Address = the address of the first register to be read. All files except the last (highest numbered) have an address range from 0 to 9999. The last file in the extended memory increment contains less than 10.000 registers. The following table lists the last address in the last file of a specified amount of extended memory

Table 3-8. Relationship Between Memory Size and Register Addresses

REGISTER ADDRESSING		
Extended Memory Size	# of Last File	Last Register Address in Last File
16K	2	6383
32K	4	2767
64K	7	5535
96K	10	8303

Extended memory is available in increments of 16K or 32K. With 32K of extended memory (32.768 accessible registers), the available register addresses of the four files will be:

File One	0 to 9999 (0000 to 270F hexadecimal)
File Two	0 to 9999 (0000 to 270F hexadecimal)
File Three	0 to 9999 (0000 to 270F hexadecimal)
File Four	0 to 2767 (0000 to 0ACF hexadecimal)

Register Count = the number of registers to be read. The maximum number of registers read is dependent upon the maximum message length. The maximum query message length is 256 bytes (including check character/s). The maximum response message length is the same. If the starting register address is 5 (0005 hexadecimal), and the number of registers to be read is 10 (000A hexadecimal), the registers read by this function would be registers 5 through 14 (10 registers).

Error Check = the longitudinal redundancy check if using ASCII transmission or the cyclic redundancy check if using RTU transmission. See paragraphs 1.2.1 and 1.2.2 for additional information on these methods of error checking.

RESPONSE

One read general reference query message can result in one or more sub-responses. The addressed slave responds with its own address, the function code, and the total byte count of one or more sub-responses. Each sub-response contains the byte count of that sub-response, its reference type, and the response data. The error check field follows the last sub-response.

SUB-RESPONSE

ADDR	FUNC	BYTE CNT	SUB RESP BYTE CNT	REF TYPE	1ST REG DATA	2ND REG DATA	NTH REG DATA
					H L	H L	H L
B	B	B	B	B	B B	B B	B B

SUB-RESPONSE

SUB RESP BYTE CNT	REF TYPE	1ST REG DATA	2ND REG DATA	NTH REG DATA	ERROR CHECK FIELD
		H L	H L	H L	
B	B	B B	B B	B B	W(CRC) OR B(LRC)

Figure 3-35. Read General Reference Response Message Format.

SUB-RESPONSE

ADDR	FUNC	BYTE CNT	SUB RESP BYTE CNT	REF TYPE	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA	5TH REG DATA
					H L	H L	H L	H L	H L
11	14	18	0B	06	0008	0002	0003	0004	0005

SUB-RESPONSE

SUB RESP BYTE CNT	REF TYPE	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA	5TH REG DATA	ERROR CHECK FIELD
		H L	H L	H L	H L	H L	
0B	06	0002	0003	0004	0005	0006	CRC OR LRC

Figure 3-36. SAMPLE: Read General Reference Response Message.

Byte Count = the total number of bytes in the read general reference response message, excluding the address, function code, byte count, and the error check fields: that is all occurences of the following: sub-response byte count, reference type, and the first through the nth register in each sub-response. Therefore in the response message in Figure 3-35, the byte count of 24 (18 hexadecimal) is the sum of the bytes from the response byte count field in the first sub-response through the fifth register in the second sub-response.

The first sub-response byte count is 11 (0B hexadecimal), and since both sub-responses happen to be the same length, the first and second added together equal a byte count of 22 (16 hexadecimal). Therefore the total byte count of the two sub-responses including the two response byte counts is 24 (18 hexadecimal).

Response Byte Count = the number of bytes in each separate sub-response. In Figure 3-35 the response byte count for the first sub-response is 11 (0B hexadecimal), as there are 10 bytes of data in the five registers read and one byte in the reference type or eleven bytes total.

The response byte count can be different for each sub-response. If, in the second sub-response, Figure 3-35, only three registers (2 bytes each) were read, the response byte count would then be 7 (07 hexadecimal).

Reference Type = must be 6 (06 hexadecimal). 584L Extended memory storage registers

1st, 2nd, 3rd ... Register Data = the information contained in each register being read. The high order byte is first, and the low order byte is second.

Error Check = the longitudinal redundancy check if using ASCII transmission or the cyclic redundancy check if using RTU transmission. See paragraphs 1.2.1 and 1.2.2 for additional information on these methods of error checking.

NOTE

Each read of extended memory includes parity error checking of the integrity of the data in memory. Refer to Table 2-1, exception response 08, for additional information about parity errors.

3.15 WRITE GENERAL REFERENCE (FUNCTION CODE 21)

QUERY

A write general reference message enters or changes information in Gould 584L extended memory files.

SUB-REQUEST

ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	→	NTH REG DATA
B	B	B	B	W	W	W	W		W

SUB-REQUEST

REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	→	NTH REG DATA	ERROR CHECK FIELD
B	W	W	W	W		W	W(CRC) OR B(LRC)

Figure 3-37. Write General Reference Query Message Format.

SUB-REQUEST

ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA
11	15	1C	06	0008	002D	0004	003C	005E	0071	0071

SUB-REQUEST

REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	ERROR CHECK FIELD
06	0009	0060	0003	0040	0041	0042	CRC OR LRC

Figure 3-38. SAMPLE: Write General Reference Query Message.

Byte Count = the total number of bytes in the write general reference message. excluding the address, function code, byte count, and the error check fields; that is all occurences of the following: reference type, file number, register address register count, and the first through the last register in the write general reference query message.

In the sample message illustrated in Figure 3-37 the byte count is 28, (1C hexadecimal) because two sub-requests are being sent, the first being 15 bytes length, and the second being 13 bytes in length.

Reference Type = must be 6 (06 hexadecimal). 584L Extended memory storage registers

File Number = the number of the file into which information is to be entered or changed. Extended memory is addressed as a group of files, each file containing up to 10,000 registers. File numbers range from 1 to 10.

Starting Register Address = the address of the first register in which information is to be entered or changed. All files except the last (highest numbered) have an address range from 0 to 9999. The last file in the extended memory increment contains less than 10,000 registers. Table 3-8 lists the last address in the last file of a specified amount of extended memory.

Register Count = the number of registers in which information will be entered or changed. The maximum number of registers in which information can be changed or entered is dependent upon the maximum message length. The maximum response message length is 256 bytes (including check characters).

If the starting register address is 5 (0005 hexadecimal), and the number of registers into which information will be written is 10 (000A hexadecimal), information will be written into registers 5 through 14 (10 registers).

1st through last Register = the information entered in each register, high order byte first, low order byte second.

Error Check = the longitudinal redundancy check if using ASCII transmission or the cyclic redundancy check if using RTU transmission. See paragraphs 1.2.1 and 1.2.2 for additional information on these methods of error checking.

RESPONSE

The normal response to a write general reference query message is the retransmission of the write request.

			SUB-RESPONSE							
ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA
11	15	1C	06	0008	002D	0004	003C	005E	0071	0071

SUB-RESPONSE							
REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	ERROR CHECK FIELD
06	0009	0060	0003	0040	0041	0042	CRC OR LRC

Figure 3-39. SAMPLE: Write General Reference Response Message.

ADDR	FUNC	BYTE CNT	SUB-REQUEST: REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	...	NTH REG DATA
B	B	B	B	W	W	W	W		W

SUB-REQUEST: REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	...	NTH REG DATA	ERROR CHECK FIELD
B	W	W	W	W		W	W(CRC) OR B(LRC)

Figure 3-37. Write General Reference Query Message Format.

ADDR	FUNC	BYTE CNT	SUB-REQUEST: REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA
11	15	1C	06	0008	002D	0004	003C	005E	0071	0071

SUB- REQUEST: REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	ERROR CHECK FIELD
06	0009	0060	0003	0040	0041	0042	CRC OR LRC

Figure 3-38. SAMPLE: Write General Reference Query Message.

Byte Count = the total number of bytes in the write general reference message. excluding the address, function code, byte count, and the error check fields; that is all occurences of the following: reference type, file number, register address register count, and the first through the last register in the write general reference query message.

In the sample message illustrated in Figure 3-37 the byte count is 28. (1C hexadecimal) because two sub-requests are being sent, the first being 15 bytes length, and the second being 13 bytes in length.

Reference Type = must be 6 (06 hexadecimal). 584L Extended memory storage registers

File Number = the number of the file into which information is to be entered or changed. Extended memory is addressed as a group of files, each file containing up to 10,000 registers. File numbers range from 1 to 10.

Starting Register Address = the address of the first register in which information is to be entered or changed. All files except the last (highest numbered) have an address range from 0 to 9999. The last file in the extended memory increment contains less than 10,000 registers. Table 3-8 lists the last address in the last file of a specified amount of extended memory.

Register Count = the number of registers in which information will be entered or changed. The maximum number of registers in which information can be changed or entered is dependent upon the maximum message length. The maximum response message length is 256 bytes (including check characters).

If the starting register address is 5 (0005 hexadecimal), and the number of registers into which information will be written is 10 (000A hexadecimal), information will be written into registers 5 through 14 (10 registers).

1st through last Register = the information entered in each register, high order byte first, low order byte second.

Error Check = the longitudinal redundancy check if using ASCII transmission or the cyclic redundancy check if using RTU transmission. See paragraphs 1.2.1 and 1.2.2 for additional information on these methods of error checking.

RESPONSE
The normal response to a write general reference query message is the retransmission of the write request.

			SUB-RESPONSE							
ADDR	FUNC	BYTE CNT	REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	4TH REG DATA
11	15	1C	06	0008	002D	0004	003C	005E	[illegible]	0071

SUB-RESPONSE							
REF TYPE	FILE NO	REG ADDR	REG CT	1ST REG DATA	2ND REG DATA	3RD REG DATA	ERROR CHECK FIELD
06	0009	0060	0003	0040	0041	0042	CRC OR LRC

Figure 3-39. SAMPLE: Write General Reference Response Message.

APPENDIX A
MODBUS TRANSACTION TIME CALCULATION

MODBUS COMMAND RESPONSE CALCULATION

The following is an analysis of the response times involved to complete a MODBUS transaction:

1. Master device composes the message

2. Query modem at master (RTS/CTS)

   (a) If the RTS and CTS pins are jumpered this time is negligible.

   (b) For the J478, the time is about 5 ms.

   (c) Times vary for other Modems.

3. Transmission of data to slave PC

$$\text{Time (ms)} = \frac{1000 \times (\text{No. of Chars.}) \times (\text{Bits Per Char.})}{(\text{Baud Rate})}$$

4. Slave processes message

   (a) Since the MODBUS command is processed during a "window" at the end of the controller scan, the worst case delay between receiving a message and commencing to process that message is one scan (i.e., the message arrives at the slave PC just after the MODBUS "window" closes). On the average, the delay would be ½ scan.

   (b) The "window" allotted to the servicing of a MODBUS command is approximately 1.5 ms. wide for the 484 and 584.

   In a 184/384, the "window" length varies depending on the amount of data requested or sent. See Figure A-1.

   Each of the programmable controllers handles its MODBUS ports in a slightly different manner.

   /1/ The 484 has its "window" shared by the J474/J475 and the J470 on a contention basis.

   /2/ The 584 has two MODBUS ports, each with its own "window". These two ports are serviced sequentially with port 1 being serviced before port 2.

   /3/ The 184/384 has a single MODBUS port. When the 184/384 is used with a programming panel, the MODBUS port is not operative and is locked out with a keyswitch.

/4/ The Micro 84 has a single Modbus port. The Modbus adapter will allow simultaneous use of both the Modbus port and the P370 programming panel port. The messages will be buffered and arbitration occurs on a first-come, first-served basis when both ports are active. In the event of a conflict, the Modbus port will be given higher priority. If the Modbus is logged in", the P370 will be restricted to monitor-only operations.

/5/ The 884 has 2 Modbus ports, one on the 884 CPU and one as an option.

In order to avoid contention only one port may have write privileges (considered as PASSCODE levels 1 through 3) at any given time. Possession of the passcode associated with a higher privilege level does not imply any priority over the ability to modify the contents of the PC: all such abilities are allocated on a first-come, first-served basis. No single Modbus function or port has a higher priority than any other. The ports are serviced in "round robin" fashion.

NOTE

However, due to timeout constraints, if a "standard" (i.e., not FC 18) Modbus function cannot be processed immediately, it is rejected with a BUSY response, FC 18 commands are retained within the CPU until they can be processed since a POLL mechanism is available.

(c) Accessing Multiple Points/Registers

Standard functions 1-4 and 15-16 permit the transfer in a single MODBUS query of more data can can be exchanged in the time allocated the responder between any two connective sweeps. Thus, it is sometimes necessary for the PC slave interface to buffer blocks of data and distribute their exchange with the slave PC memory over multiple consecutive sweeps.

The amount of data that can be processed during a single MODBUS "window" varies according to the type of controller used. Table A-1 shows the quantities allowed.

Table A-1. Data Quantities for Single Modbus "Window"

PC Type	Discretes	Registers
Micro 84	16	4
484	32	16
584	64	32
184/384	800	100
884	*	*See Note 1

If a transaction requires more data than these limits allow, the controller will continue to handle the maximum amount of data possible during each "window" until it finishes processing all of the data. For example a request to read 80 registers in a 584 would require 3 "windows" to finish gathering this data: window 1 would gather 32 registers, window 2 would gather 32 registers, and window 3 would get the remaining 16 registers. The transmission of the response frame is not initiated until all data is assembled.

NOTE No. 1

For both standard Modbus Commands and Programming Subfunctions in the 884 and Micro 84 affecting references (either registers or discretes) all data is read from or to state RAM during the space of 1 scan. (The design of the 884 allows a maximum scan impact of 2 ms per scan for Modbus message handling.) However, validation and setup and data will require several time slices to be accomplished. For typical latency times, see Note 2 of the 884.

NOTE No. 2

Latency time is defined as the time starting when the Modbus message is received by the port until the bit stream is transmitted to the Modbus port or state table changed. These times vary for each function code. Timing tests for typical function codes in the 884 are:

FC 1	read 768 coils	14 scans
FC 2	read 256 inputs	7 scans
FC 3	read 125 output registers	5 scans
FC 4	read 125 input registers	8 scans
FC 5	force coil	3 scans
FC 6	preset registers	3 scans
FC 15	force multiple (768) coils	18 scans
FC 16	preset multiple (100) reg.	10 scans

5. Slave PC performs error check calculation.

   (a) LRC calculation time is less than 1 ms.

   (b) CRC calculation time is 0.3 ms. for each 8 bits of data in the response.

6. Query modem at slave PC (RTS/CTS).

   (a) J478 and J474 modems require about 5 ms.

   (b) Times vary for other modems.

   (c) RTS and CTS pins may not be jumpered at this end.

7. Transmission of response to master.

$$\text{Time (ms)} = \frac{1000 \times (\text{No. of Chars.}) \times (\text{Bits Per Char.})}{(\text{Baud Rate})}$$

8. Master receives response, verifies error check, strips out data, and passes data to the applications program.

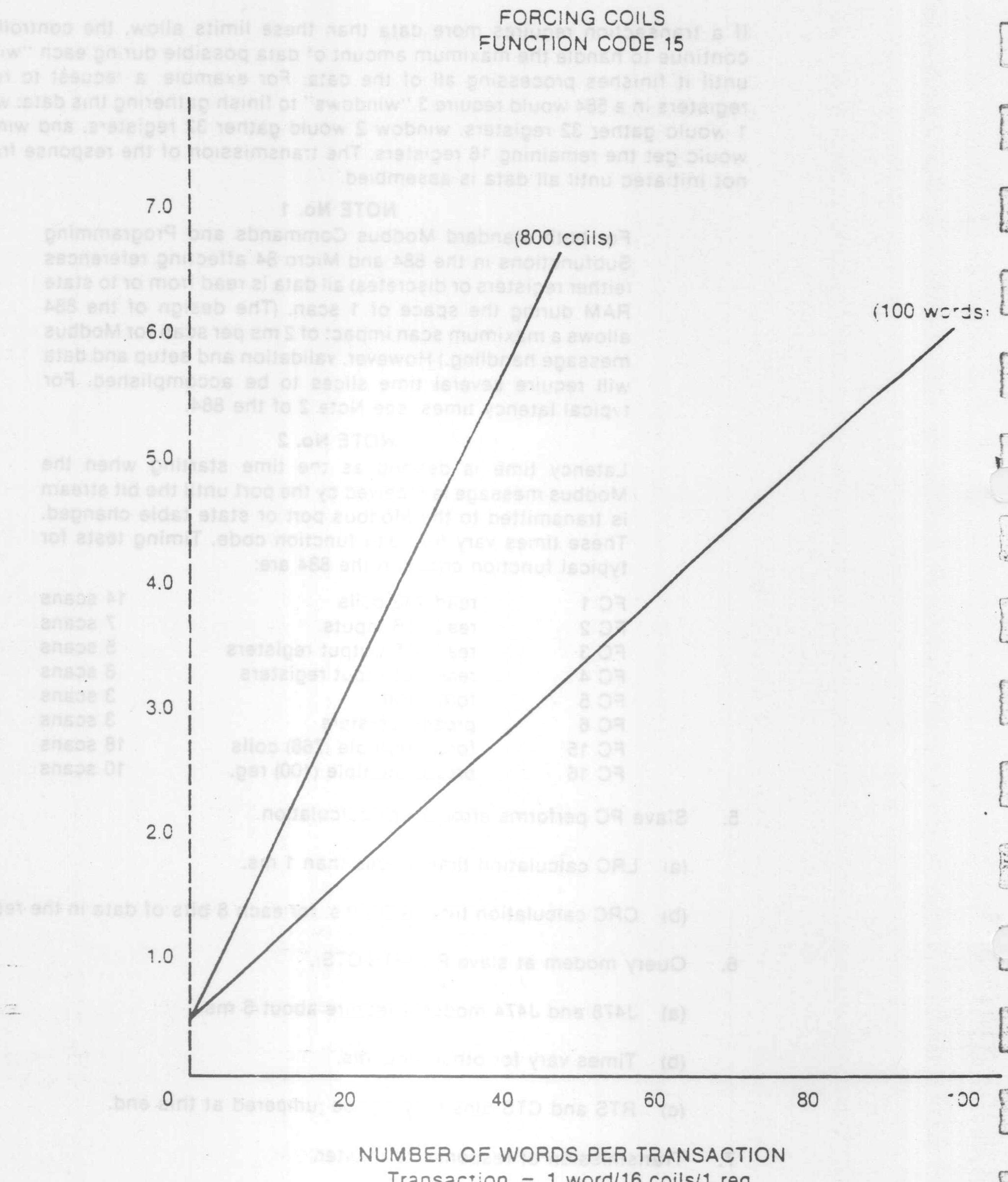

Figure A-1. Impact of Modbus Commands on 184/384 Scan Time

APPENDIX B
MAXIMUM QUERY AND RESPONSE DATA PARAMETERS FOR 184/384, 484, MICRO 84, 884, AND 584 CONTROLLERS

Table B-1. Maximum Query and Response Data Parameters for 184/384 Controllers

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
1	READ COIL STATUS	800 COILS	800 COILS
2	READ INPUT STATUS	800 INPUTS	800 INPUTS
3	READ HOLDING REG.	100 REGISTERS	100 REGISTERS
4	READ INPUT REG.	100 REGISTERS	100 REGISTERS
5	FORCE COIL	1 COIL	1 COIL
6	LOAD REGISTER	1 REGISTER	1 REGISTER
7	READ EXCEPTION STATUS	N/A	8 COILS
8	LOOP BACK DIAGNOSTIC	N/A	N/A
9	PROGRAM 484	NOT SUPPORTED	NOT SUPPORTED
10	POLL 484	NOT SUPPORTED	NOT SUPPORTED
11	COMM. EVENT COUNTER	N/A	N/A
12	COMM. EVENT LOG	N/A	70 DATA BYTES
13	PROGRAM-GENERAL	32 DATA BYTES	32 DATA BYTES
14	POLL-GENERAL	N/A	32 DATA BYTES
15	FORCE MULTIPLE COILS	800 COILS	800 COILS
16	LOAD MULTIPLE REGS.	100 REGISTERS	100 REGISTERS
17	REPORT SLAVE I.D.	N/A	N/A
18	PROGRAM	NOT SUPPLIED	NOT SUPPLIED
19	RESET COMMUNICATIONS LINK	NOT SUPPLIED	NOT SUPPLIED

Table B-2 Maximum Query and Response Data Parameters for 484 Controllers

Values are shown for an 8K controller. See the Modbus 484 Programming Protocol Reference Guide or the Gould 484 User's Guide for maximum sizes of smaller controllers.

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
1	READ COIL STATUS	512 COILS	512 COILS
2	READ INPUT STATUS	512 INPUTS	512 INPUTS
3	READ MOLDING REG.	254 REGISTERS	254 REGISTERS

Table B-2. Maximum Query and Response Data Parameters for 484 Controllers (cont.)

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
4	READ INPUT REG.	32 REGISTERS	32 REGISTERS
5	FORCE COIL	1 COIL	1 COIL
6	LOAD REGISTER	1 REGISTER	1 REGISTER
7	READ EXCEPTION STATUS	N/A	8 COILS
8	LOOP BACK DIAGNOSTIC	N/A	N/A
9	PROGRAM 484	16 DATA BYTES	16 DATA BYTES
10	POLL 484	N/A	16 DATA BYTES
11	COMM. EVENT COUNTER	NOT SUPPORTED	NOT SUPPORTED
12	COMM. EVENT LOG	NOT SUPPORTED	NOT SUPPORTED
13	PROGRAM-GENERAL	NOT SUPPORTED	NOT SUPPORTED
14	POLL-GENERAL	NOT SUPPORTED	NOT SUPPORTED
15	FORCE MULTIPLE COILS	800 COILS	800 COILS
16	LOAD MULTIPLE REGS.	60 REGISTERS	60 REGISTERS
17	REPORT SLAVE I.D.	N/A	N/A
18	PROGRAM	NOT SUPPORTED	NOT SUPPORTED
19	RESET COMMUNICATIONS LINK	NOT SUPPORTED	NOT SUPPORTED

Table B-3. Maximum Query and Response Data Parameters for 584 Controllers

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
1	READ COIL STATUS	2000 COILS	2000 COILS
2	READ INPUT STATUS	2000 INPUTS	2000 INPUTS
3	READ HOLDING REG.	125 REGISTERS	125 REGISTERS
4	READ INPUT REG.	125 REGISTERS	125 REGISTERS
5	FORCE COIL	1 COIL	1 COIL
6	LOAD REGISTER	1 REGISTER	1 REGISTER
7	READ EXCEPTION STATUS	N/A	8 COILS
8	LOOP BACK DIAGNOSTIC	N/A	N/A
9	PROGRAM 484	NOT SUPPORTED	NOT SUPPORTED
10	POLL 484	NOT SUPPORTED	NOT SUPPORTED
11	COMM. EVENT COUNTER	N/A	N/A

Table B-3. Maximum Query and Response Data Parameters for 584 Controllers (cont.)

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
12	COMM. EVENT LOG	N/A	70 DATA BYTES
13	PROGRAM-GENERAL	33 DATA BYTES	33 DATA BYTES
14	POLL-GENERAL	N/A	33 DATA BYTES
15	FORCE MULTIPLE COILS	800 COILS	800 COILS
16	LOAD MULTIPLE REGS.	100 REGISTERS	100 REGISTERS
17	REPORT SLAVE I.D.	N/A	N/A
18	PROGRAM	NOT SUPPORTED	NOT SUPPORTED
19	RESET COMMUNICATIONS LINK	NOT SUPPORTED	NOT SUPPORTED
20	READ GENERAL REFERENCE	*	*
21	WRITE GENERAL REFERENCE	**	**

* Refer to Section 3.14 for information.
** Refer to Section 3.15 for information.

Table B-4. Maximum Query and Response Data Parameters for Micro 84 Controllers

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
1	READ COIL STATUS	64 COILS	64 COILS
2	READ INPUT STATUS	64 INPUTS	64 INPUTS
3	READ HOLDING REG.	32 REGISTERS	32 REGISTERS
4	READ INPUT REG.	4 REGISTERS	4 REGISTERS
5	FORCE COIL	1 COIL	1 COIL
6	LOAD REGISTER	1 REGISTER	1 REGISTER
7	READ EXCEPTION STATUS	N/A	8 COILS
8	LOOP BACK DIAGNOSTIC	N/A	N/A
9	PROGRAM 484	NOT SUPPORTED	NOT SUPPORTED
10	POLL 484	NOT SUPPORTED	NOT SUPPORTED
11	COMM. EVENT COUNTER	NOT SUPPORTED	NOT SUPPORTED
12	COMM. EVENT LOG	NOT SUPPORTED	NOT SUPPORTED
13	PROGRAM-GENERAL	NOT SUPPORTED	NOT SUPPORTED

MAXIMUM QUERY AND RESPONSE DATA PARAMETERS

Table B-4. Maximum Query and Response Data Parameters for Micro 84 Controllers (cont.)

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
14	POLL-GENERAL	NOT SUPPORTED	NOT SUPPORTED
15	FORCE MULTIPLE COILS	64 COILS	64 COILS
16	LOAD MULTIPLE REGS.	32 REGISTERS	32 REGISTERS
17	REPORT SLAVE I.D.	N/A	N/A
18	PROGRAM	*	*
19	RESET COMMUNICATIONS LINK	N/A	N/A

*The total number of Modbus message sent or received by the Micro 84 cannot exceed the buffer size. Each subfunction has different limits.

Table B-5. Maximum Query and Response Data Parameters for 884 Controllers

The values shown are theoretical based on the buffer size of the Modbus port of the 884. The actual values may be limited by the configuration of the individual 884 (the 884-1 will typically be less). See the 884 (CONFIG-SYS PLAN GUIDE) for configuration limitations.

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
1	READ COIL STATUS	2000 COILS	2000 COILS
2	READ INPUT STATUS	2000 INPUTS	2000 INPUTS
3	READ HOLDING REG.	125 REGISTERS	125 REGISTERS
4	READ INPUT REG.	125 REGISTERS	125 REGISTERS
5	FORCE COIL	1 COIL	1 COIL
6	LOAD REGISTER	1 REGISTER	1 REGISTER
7	READ EXCEPTION STATUS	N/A	8 COILS
8	LOOP BACK DIAGNOSTIC	N/A	N/A
9	PROGRAM 484	NOT SUPPORTED	NOT SUPPORTED
10	POLL 484	NOT SUPPORTED	NOT SUPPORTED
11	COMM. EVENT COUNTER	NOT SUPPORTED	NOT SUPPORTED
12	COMM. EVENT LOG	NOT SUPPORTED	NOT SUPPORTED
13	PROGRAM-GENERAL	NOT SUPPORTED	NOT SUPPORTED

Table B-5. Maximum Query and Response Data Parameters for 884 Controllers (cont.)

FUNCTION CODE	DESCRIPTION	QUERY	RESPONSE
14	POLL-GENERAL	NOT SUPPORTED	NOT SUPPORTED
15	FORCE MULTIPLE COILS	800 COILS	800 COILS
16	LOAD MULTIPLE REGS.	100 REGISTERS	100 REGISTERS
17	REPORT SLAVE I.D.	N/A	N/A
18	PROGRAM	*	* SEE NOTE
19	RESET COMMUNICATIONS LINK	N/A	N/A

NOTE

Total number of Modbus messages cannot exceed the buffer limit of 256 bytes. Each subfunction has different limits.

APPENDIX C
IMPLIED LENGTH SUMMARY

FUNCTION CODE	QUERY IMPLIED LENGTH LESS ERROR CHECK FIELD*	RESPONSE IMPLIED LENGTH LESS ERROR CHECK FIELD*
0	Not Defined	Not Defined
1	6	3 + 3rd Byte
2	6	3 + 3rd Byte
3	6	3 + 3rd Byte
4	6	3 + 3rd Byte
5	6	6
6	6	6
7	2	3
8	6	6
9	2 + 5th Byte	2 + 5th Byte
10	2	2 + 5th Byte
11	2	3 + 3rd Byte
12	2	3 + 3rd Byte
13	3 + 3rd Byte	3 + 3rd Byte
14	2	3 + 3rd Byte
15	7 + 7th Byte	6
16	7 + 7th Byte	6
17	2	3 + 3rd Byte
18	3 + 3rd Byte	3 + 3rd Byte
19	3 + 3rd Byte	3 + 3rd Byte
20	2 + 3rd Byte	2 + 3rd Byte
21	2 + 3rd Byte	2 + 3rd Byte
22-126	3 + 3rd Byte	3 + 3rd Byte
127	2	2
128-255	Not Defined	3

*To obtain total implied frame length, add 2 bytes if RTU (for CRC-16) or add 1 byte if ASCII (for LRC). See example on next page.

NOTE

The number of characters used to transmit the frame differs for RTU and ASCII (see Section 3).

For RTU, number of characters equals number of bytes.

For ASCII, number of characters equals twice the number of bytes plus 3 (colon, CR, and LF).

IMPLIED LENGTH EXAMPLE:

In the below example register 4136 is set to 10 and register 4137 is set to 258.

ADDR	FUNC	HO ADDR	LO ADDR	QUANTITY		BYTE #7 CNT	HO DATA	LO DATA	HO DATA	LO DATA	
30	10	00	87	00	02	04	00	0A	01	02	ERROR CHECK FIELD

Figure C-1. Preset Multiple Registers Query Message

RESPONSE

ADDR	FUNC	HO ADDR	LO ADDR	QUANTITY		ERROR CHECK FIELD
30	10	00	87	00	02	

Figure C-2. Preset Multiple Registers Response Message

For the query in Figure C-1, the implied length table indicates that the byte count is 7 + the 7th byte. The 7th byte of the query is 04, giving a total length of 7 + 4 = 11 bytes not including the error check bytes (2 for RTU and 1 for ASCII) which vary depending on the communication mode.

For the response in Figure C-2, the implied length table indicates that the byte count is 6 (not including the error check bytes) which agrees with the response shown.